AF

Docket No.: 1046.1236

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Yutaka HAGA

Serial No. 09/778,076 Group Art Unit: 2192

Confirmation No. 4573

Filed: February 7, 2001 Examiner: Yigdall, Michael J.

For: APPARATUS FOR COLLECTING PROFILES OF PROGRAMS

COMMUNICATION IN RESPONSE TO NOTIFICATION OF NON-COMPLIANT APPEAL BRIEF

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

Attached is a Substitute Appeal Brief in response to the Notification of Non-Compliant Appeal Brief mailed February 13, 2007. In the Notification, box 10 was checked indicating that the incorrect serial number was in the upper right hand corner of the pages. In the attached Substitute Appeal Brief, the serial number is appropriately corrected.

Respectfully submitted,

STAAS & HALSEY LLP

Date: February 20, 2007 By: _____

Paul W. Bobowiec
Registration No. 47,431

1201 New York Avenue, NW, 7th Floor
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

Docket No. 1046.1236

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Yutaka HAGA

Application No.: 09/778.076 Group Art Unit: 2192

Filed: February 7, 2001 Examiner: Yigdall, Michael J.

For: APPARATUS FOR COLLECTING PROFILES OF PROGRAMS

SUBSTITUTE APPEAL BRIEF

Mail Stop Appeal Brief-Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In response to a Notification of Non-Compliant Appeal Brief mailed on February 13, 2007, a Substitute Appeal Brief is submitted herewith.

I. REAL PARTY IN INTEREST

The real party in interest is Fujitsu Limited, the assignee of the subject application.

II. RELATED APPEALS AND INTERFERENCES

Appellant's legal representative is not aware of any prior or pending appeals or interferences which directly affect or are directly affected by, or have a bearing, on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 8-17, 19-28 and 30-42 are pending.

Claims 8-17, 19-28, and 30-42 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Alexander, III et al. (U.S.P. 6,002,872) (Alexander) in view of Smolders (U.S.P. 6,253,338) (Smolders) and Yeh et al. (U.S.P. 6,427,206) (Yeh).

IV. STATUS OF AMENDMENTS

No amendment(s) has been filed subsequent to the final rejection made on July 12, 2006.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed invention in independent claim 12 recites an apparatus for collecting a profile of a subroutine included in a program (see, for example, FIG. 1 and paragraph 24). The apparatus of claim 12 includes a storage unit (see, for example, FIG. 2B and control table 18, paragraph [0060]) storing a profile and an analyzing section 16 (see, for example, FIG. 2B and paragraph [0051]). The apparatus of claim 12 also recites an analyzing section, when an interrupt is generated by execution of a branch instruction during execution of the program, obtaining a branch source address and a branch destination address from a source of the interrupt (see, for example, FIG. 2B and paragraph [0053], and FIG. 4 and S1 and S2, and paragraphs [0092]- [0093]), and identifying a type of the branch instruction by obtaining an instruction code from the branch source address and decoding the instruction code (see, for example, FIG. 2B and paragraph [0053], and FIG. 4 and S3 and S4, and paragraphs [0092]-[0094]).

The apparatus of claim 12 also includes a collecting section 17 (see, for example, FIG. 2B and paragraph [0057]) obtaining the branch source address, the branch destination address, and an identified result from the analyzing section (see, for example, FIG. 4 S5 and paragraph [0094]) when the identified instruction is a calling instruction or a return instruction of the subroutine (see, for example, FIG. 2B and paragraph [0057]). The apparatus of claim 12 also includes a collecting section 17 when the identified result is the calling instruction, storing the branch destination address as a subroutine address corresponding to the calling instruction and a calling time of the subroutine corresponding to the calling instruction in the storage unit (see, for example, FIG. 2B and control table 18, paragraph [0060] and FIG. 5 and paragraphs [0098]-[0099]).

The apparatus of claim 12 also includes a collecting section 17 when the identified result is the return instruction, obtaining a return time of the subroutine corresponding to the return instruction, calculating a execution time of the subroutine based on the obtained return time and the calling time (see, for example FIG. 7 and paragraphs [0118] and [0119]), and storing a cumulative value of the execution time as the profile in correspondence with the branch destination address in the storage unit (see, for example, FIG. 2B and control table 18, paragraph [0060] and FIG. 7 and paragraphs [0121]-[0123]).

The apparatus of claim 12 includes when the identified branch instruction is neither a calling instruction nor a return instruction, the interrupt is terminated (see, for example, FIG. 4 and S6 and paragraph [0094]). The apparatus of claim 12 also includes a collecting section 17

when the identified result is the calling instruction, storing the branch destination address as a subroutine address corresponding to the calling instruction and a calling time of the subroutine corresponding to the calling instruction in the storage unit (see, FIG. 2B and control table 18, and paragraph [0060], and FIG. 7 and paragraph [0133]).

The claimed invention in independent claim 23 recites a computer readable medium storing a program for a computer executing a process for collecting a profile of a subroutine included in a target program (see, for example, FIG. 1 and paragraph [0024] and FIG. 4, FIG. 5, FIG. 6, and FIG. 7). The computer readable medium of claim 23 includes a program obtaining a branch source address and a branch destination address from a source of an interrupt when the interrupt is generated by execution of a branch instruction during execution of the target program see, for example, FIG. 4 and S1 and S2, and paragraphs [0092]- [0093]).

The computer readable medium of claim 23 includes a program identifying a type of the branch instruction by obtaining an instruction code from the branch source address and decoding the instruction code (see, for example, FIG. 4 and S3 and S4, and paragraph [0092] - [0094]). The computer readable medium of claim 23 includes a program storing the branch destination address as a subroutine address corresponding to the calling instruction and a calling time of the subroutine corresponding to the calling instruction in a storage unit (see, for example, FIG. 2B and control table 18, and paragraph [0060]) when the identified result is the calling instruction.

The computer readable medium of claim 23 includes a program when the identified result is the return instruction, obtaining a return time of the subroutine corresponding to the return instruction and calculating a execution time of the subroutine based on the obtained return time and the calling time (see, for example FIG. 7 and paragraphs [0118] and [0119]). The computer readable medium of claim 23 includes a program storing a cumulative value of the execution time as the profile in correspondence with the branch destination address in the storage unit (see, for example, FIG. 2B and control table 18, and paragraph [0060]).

The claimed invention in claim independent claim 23 includes a program when the identified branch instruction is neither a calling instruction nor a return instruction, the interrupt is terminated (see, for example, FIG. 4 and S6 and paragraph [0094]).

The claimed invention in independent claim 34 recites a method for collecting a profile of a subroutine included in a program (see, for example FIG. 4, FIG. 5., FIG. 6, and FIG. 7). The method of claim 34 includes obtaining a branch source address and a branch destination address from a source of an interrupt when the interrupt is generated by execution of a branch

instruction during execution of the program (see, for example, FIG. 4 and S1 and S2, and paragraphs [0092]- [0093]) and identifying a type of the branch instruction by obtaining an instruction code from the branch source address and decoding the instruction code (see, for example, FIG. 4 and S3 and S4, and paragraph [0092] - [0094]). The claimed invention in independent claim 34 recites a method storing the branch destination address as a subroutine address corresponding to the calling instruction and a calling time of the subroutine corresponding to the calling instruction in a storage unit (see, for example, FIG. 2B and control table 18, paragraph [0060]) when the identified result is the calling instruction (see, for example, FIG. 5 and paragraphs [0098]-[0099]).

The claimed invention in independent claim 34 recites a method when the identified result is the return instruction, obtaining a return time of the subroutine corresponding to the return instruction and calculating a execution time of the subroutine based on the obtained return time and the calling time (see, for example FIG. 7 and paragraphs [0118] and [0119]). The claimed invention in independent claim 34 recites a method storing a cumulative value of the execution time as the profile in correspondence with the branch destination address in the storage unit (see, for example, FIG. 2B and control table 18, paragraph [0060] and FIG. 7 and paragraphs [0121]-[0123]).

The claimed invention in independent claim 34 recites a method when the identified branch instruction is neither a calling instruction nor a return instruction, the interrupt is terminated (see, for example, FIG. 4 and S6 and paragraph [0094]).

VI. *GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL*

Claims 8-17, 19-28, and 30-42 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Alexander in view of Smolders and Yeh.

Claims are each independently patentable over the references as set forth below, and do not stand or fall together.

VII. ARGUMENT

All arguments are directed to the grounds of rejection. All citations to the "Office Action" refer to the last and final Office Action of July 12, 2006.

In the Office Action, the Examiner rejects claims 8-17, 19-28, and 30-42 under 35 U.S.C. §103(a) as being unpatentable over Alexander in view of Smolders and Yeh.

An issue is whether Alexander in view of Smolders and Yeh suggest to one skilled in the art to achieve, the recited features of claims 8-17, 19-28, and 30-42.

A first sub-issue is whether the Examiner has established a *prima facie* case of obviousness based upon Alexander in view of Smolders and Yeh.

A second sub-issue is whether the Examiner has provided evidence which as a whole show that the legal determination sought to be proved (i.e., whether the reference teachings establish a *prima facie* case of obviousness) is more probable than not by the preponderance of evidence burden-of-proof standard (37 CFR 1.56(b)).

A. The Law Regarding the Obviousness Issues

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ 2d 1596 (Fed. Cir. 1988). See Manual Of Patent Examining Procedure § 2143.03 (8th ed. Rev. 5 August 2006)("MPEP").

Impermissible hindsight must be avoided and the legal conclusion must be reached on the basis of the facts gleaned from the prior art. See MPEP § 2142 Legal Concept of *Prima Facie* Obviousness.

An assessment of basic knowledge and common sense that is not based on any evidence in the record lacks substantial evidence support. Id. at 1385, 59 USPQ 2d at 1697. See also *In re Lee*, 277 F.3d 1338, 1344-45, 61 USPQ 2d 1430, 1434-35 (Fed. Cir. 2002) (In reversing the Board's decision, the court stated "'common knowledge and common sense' on which the Board relied in rejecting Lee's application are not the specialized knowledge and expertise contemplated by the Administrative Procedure Act. Conclusory statements such as those here provided do not fulfill the agency's obligation...The board cannot rely on conclusory statements

when dealing with particular combinations of prior art and specific claims, but must set forth the rationale on which it relies."). See MPEP § 2144.03.

B. Errors in Examiner's Contention

1) The Prior Art

Alexander is directed to a method for monitoring the performance of an entire system with one or more active programs. A periodically occurring event is identified and the call stack associated with the active thread at the time of the event is obtained. The call stack for the thread is examined to identify each routine currently executing. Each routine in the call stack is represented as a node in an accounting data structure having the organization of a tree. See, for example, col. 2, lines 38-46.

Smolders is directed to a method and system within a data processing system or information handling system for counting various events from a running program by taking a trace by way of using an interruption. A processor within a data processing system is programmed to generate a trace interrupt at least after each branch instruction, or at the end of each basic block of code from a currently running program or process. By programming monitor mode control registers within a performance monitor feature, one or more counters are programmed to count various events happening on the data processing system thereby creating tracing information. See, for example, col. 2, lines 2-38.

Yeh is directed to a microprocessor including a branch prediction table (BPT) that has at least one branch entry. The at least one branch entry includes a prediction field to indicate whether a branch is predicted taken. The at least one branch entry also includes a history register to store history information. See, for example, col. 2, lines 17-24.

2) Recited Features Not Discussed By Cited Art And *Prima Facie* Obviousness Not Established

a. Claim 12

Claim 12 recites an apparatus for collecting a profile of a subroutine included in a program, comprising: "a storage unit storing a profile; an analyzing section, when an interrupt is generated by execution of a branch instruction during execution of said program: obtaining a branch source address and a branch destination address from a source of said interrupt, and identifying a type of said branch instruction by obtaining an instruction code from said branch source address and decoding said instruction code; and a collecting section: obtaining said branch source address, said branch destination address, and an identified result from said analyzing section when the identified instruction is a calling instruction or a return instruction of

said subroutine; and when said identified result is said calling instruction, storing said branch destination address as a subroutine address corresponding to said calling instruction and a calling time of said subroutine corresponding to said calling instruction in said storage unit, when said identified result is said return instruction, obtaining a return time of said subroutine corresponding to said return instruction, calculating a execution time of said subroutine based on said obtained return time and said calling time, and storing a cumulative value of said execution time as said profile in correspondence with said branch destination address in said storage unit, and when the identified branch instruction is neither a calling instruction nor a return instruction, said interrupt is terminated."

Appellant submits that recited features of a method "identifying a type of said branch instruction by obtaining an instruction code from said branch source address and decoding said instruction code; and . . . and when the identified branch instruction is neither a calling instruction nor a return instruction, said interrupt is terminated" are not discussed by an *arguendo* combination of Alexander in view of Smolders and in view of Yeh.

The Office Action concedes that:

> Alexander does not expressly disclose the limitation wherein the interrupt is generated by execution of a branch instruction, and Alexander does not expressly disclose identifying a type of said branch instruction by obtaining an instruction code from said branch source address and decoding said instruction code.

(Office Action at page 6, lines 8-11).

The Examiner relies on the teachings of Yeh:

> [T]o supplement the profiling system of Alexander . . . and to identify the type of branch, as taught by Yeh, so as to collect branch predictions and enhance the collection of profiles for purposes of speculative execution.

(Office Action at page 7, lines 14- 17).

(1) Examiner's statements regarding "interrupts" do not properly support an establishment of *prima facie* obviousness

In support of the rejection, the Examiner asserts:

> Alexander discloses generating timer interrupts. . . and suggests that other interrupts may be generated instead . . . Smolders discloses generating an interrupt by execution of a branch instruction . . . (and) an instruction flow unit that dispatches instructions to selected execution units for execution . . . Smolders discloses a system for collecting a trace of a program . . . wherein an interrupt is generated by execution of a branch instruction, and wherein the type of the branch instruction is inherently identified by obtaining an instruction code and decoding the instruction code. The system enables tracing without introducing any overhead and without modifying the code.

(Emphasis added, Office Action at page 6, line 6 - page 7, line 2).

The Examiner asserts:

> It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the profiling system of Alexander with the features taught by Smolders and to substitute the timer interrupt of Alexander with the branch interrupt of Smolders, as suggested by Alexander, so as to obviate any overhead and modifications to the code.

(Emphasis added, Office Action at page 7, lines 3- 6).

Appellant submits the Examiner has not provided required support for the statement that it is obvious to "substitute the timer interrupt of Alexander with the branch interrupt of Smolders" and the statement is, rather, a conclusory statement that does not properly support an establishment of *prima facie* obviousness.

While Alexander does discuss (see, for example, col. 11, lines 23-25) that "other interrupts may be used to trigger the described sampling mechanism," on the other hand Alexander clearly discusses:

> (a) sample is obtained each time a periodic event, such as, for example, a timer interrupt occurs (emphasis added).

(See, for example, col. 5, lines 35-40).

That is, Alexander clearly distinguishes, for one of ordinary skill in the art a difference between different types of interrupts, in that a result "sample is obtained" for, e.g., an interrupt that is a "periodic event.

In addition, Smolders discusses generation of the trace interrupt as:

> [T]he instruction flow unit 26 is programmed to generate a trace interrupt after each branch by setting a specified branch trace enable bit 80 in the machine state register 76. . . . the equivalent of the BE bit in the MSR and an automatic trace interruption after each branch, the processor would be programmed in single-step mode, i.e. to generate a trace interruption after each instruction. In such a case, the code handling the trace interruptions will have to check for branch instructions. When a branch is found, the flow of execution proceeds as described below in FIG. 3, if the instruction was not a branch, the flow of execution simply returns to the next instruction in sequence without any additional action.

(See, for example, cols. 3, line 58 - col. 4, line 6).

That is, Smolders does not describe the trace interrupt as a periodic event.

Appellant submits that it is not obvious for one or ordinary skill in the art to have replaced a periodic event, such as, for example, a timer interrupt with a "trace interrupt" that may not be periodic. Accordingly, the Examiner has not supported his statement that Alexander's discussion of a periodic event, such as, for example, a timer interrupt is replaceable with a "trace interrupt" as taught by Smolders, and *prima facie* obviousness is not established.

In the previous Amendment filed on May 9, 2006, the Appellant presented the argument that a similar Examiner's statement (in the previous Office Action mailed January 9, 2006 that Alexander's discussion of a periodic event, such as a timer interrupt is replaceable with a "trace interrupt" as taught by Smolders) was not supported.

In item 2 of the Office Action, the Examiner responds to these arguments and asserts:

> [T]he timer interrupts are not the only "periodic events" that Alexander contemplates. If one may substitute the timer interrupts with page fault interrupts, as Alexander suggests, certainly one may also substitute the timer interrupts with other interrupts, as Alexander also suggests, such as with the trace interrupts of Smolders.

(Emphasis added, Office Action at page 3).

Appellant submits that the Examiner's statement that since Alexander *arguendo* substitutes a timer interrupt with a page fault interrupt than "certainly one may also substitute the timer interrupts with other interrupts" is conclusory and not supported.

Further, Appellant submits features the Examiner asserts that Alexander merely "contemplates," but not discussed, are not obvious to one or ordinary skill in the art. Appellant submits that such contemplations do not support establishing *prima facie* obviousness.

(2) Examiner's statements regarding "return address" do not properly support an establishment of *prima facie* obviousness

In item 2 of the Office Action, the Examiner asserts Smolders teaches:

> "the address of the beginning of the next basic block of code" is indeed a return address. Smolders expressly illustrates, "Next Block = Return Address" (step 34 in FIG. 3). The interrupt returns to "the address where the interruption came from."

(Emphasis added, Action at pages 2-3).

However, Smolder, in detail, discusses that:

> counter level tracing tool 31 saves the address of the beginning of the next basic block of code, which is the address where the interruption came from as shown in step 34"

(Emphasis added, col. 4, lines 31-35).

That is, Smolder does not teach such a feature "so as to provide a return address," as the Examiner asserts in support of the rejection.

(3) Summary

Even an *arguendo* combination of Alexander, Smolders, and Yeh does not explicitly disclose the claimed subject matter. Further, *prima facie* obviousness is not established, since the Office Action does not provide articulated reasoning with rational underpinning to support the legal

conclusion of obviousness, based upon an implicit showing of what the combined teachings, knowledge of one of ordinary skill in the art, and the nature of the problem to be solved as a whole would have suggested to those of ordinary skill in the art. Instead the Examiner has provided conclusory statements without being based on evidence of fact.

Accordingly, the Office Action has not provided evidence which as a whole show that the legal determination sought to be proved (i.e., whether Alexander in view of Smolders and Yeh establish obviousness) is more probable than not by the preponderance of evidence burden-of-proof standard (37 CFR 1.56(b).

Thus, it is submitted that the rejection of claim 12 is correct and the rejection of claim 12 should be reversed.

b. Claim 23

Independent claim 23 recites a computer readable medium storing a program for a computer executing a process for collecting a profile of a subroutine included in a target program, by: "identifying a type of said branch instruction by obtaining an instruction code from said branch source address and decoding said instruction code; . . .and when the identified branch instruction is neither a calling instruction nor a return instruction, said interrupt is terminated."

Even an *arguendo* combination of Alexander, Smolders, and Yeh does not explicitly disclose the claimed subject matter, and further, *prima facie* obviousness is not established for the reasons discussed herein in conjunction with claim 12. Therefore, it is submitted that claim 23 patentably distinguishes over the prior art.

Thus, the rejection of claim 23 is incorrect and it is submitted that the rejection of claim 23 should be reversed.

c. Claim 34

Independent claim 34 recites a method for collecting a profile of a subroutine included in a program, comprising: "identifying a type of the branch instruction by obtaining an instruction code from the branch source address and decoding the instruction code; . . . when the identified branch instruction is neither a calling instruction nor a return instruction, said interrupt is terminated."

Even an *arguendo* combination of Alexander, Smolders, and Yeh does not explicitly disclose the claimed subject matter, and further, *prima facie* obviousness is not established for the reasons discussed herein in conjunction with claim 12. Therefore, it is submitted that claim 34 patentably distinguishes over the prior art.

Thus, the rejection of claim 34 is incorrect and it is submitted that the rejection of claim 34 should be reversed.

d. Claim 40

Dependent claim 40 recites an apparatus according to claim 13, which is dependent on claim 12 wherein "the collecting section generates a control table corresponding to each executor of the subroutine on the storage unit, wherein the control table includes an executor managing table, a subroutine managing table, and a calling managing table, wherein the executor managing table stores an identifier of the executor and a pointer to assign the subroutine managing table, wherein the subroutine managing table is generated for every subroutine executed by the executor, the subroutine managing table storing a subroutine address, times of calling of the subroutine, a cumulative execution time of the subroutine, the last called time of the subroutine, and a pointer to assign the calling managing table, and wherein the calling managing table is generated for every subroutine called by the subroutine, the calling managing table storing a branch source address as a calling subroutine address, a branch destination address as a called subroutine address, times of calling of the called subroutine, a cumulative execution time of the called subroutine, the last called time of the called subroutine, and a pointer to specify the subroutine managing table managing the calling subroutine."

Appellant submits that none of the cited art discusses "an executor managing table, a subroutine managing table, and a calling managing table."

The Office Action concedes that:

> Alexander does not expressly name or label these elements the "subroutine managing table," and does not expressly disclose a "pointer to assign the subroutine managing table."

(Office Action at page 4, lines 20-22).

However, the Examiner asserts:

> The rejection is not based solely on the knowledge of the examiner, but rather on Alexander's suggestion. . .Alexander even suggests that "other pointers may be stored within the nodes to further aid subsequent analysis," as noted above. Furthermore, the names or labels given to the tables, as recited in the claims, do not render them functionally distinct from Alexander's data structures. Therefore, absent any evidence of new or unexpected results, it would have been obvious to one of ordinary skill in the art at the time the invention was made to include such tables in pointers in Alexander.

(Emphasis added, Office Action at page 4, line 3 - page 5, line 4).

Appellant submits that claim 40 recites, specific relationships, for example, in that the "executor managing table" stores "an identifier of the executor and a pointer to assign the

subroutine managing table" (emphasis added). As another example, claim 40 recites, a specific relationship of a calling managing table stores "a branch source address as a calling subroutine address, a branch destination address as a called subroutine address, times of calling of the called subroutine, a cumulative execution time of the called subroutine, the last called time of the called subroutine, and a pointer to specify the subroutine managing table managing the calling subroutine (emphasis added)."

Appellant submits that the Examiner's statement that "names or labels given to the tables, as recited in the claims, do not render them functionally distinct from Alexander's data structures is incorrect.

That is, the specific recited names and labels given to the tables in a certain relationship, as recited in claim 40 do render them "functionally distinct" from an arbitrary association of pointers and nodes referred to by the Examiner.

Since *prima facie* obviousness is not established, it is submitted the rejection of claim 40 is incorrect and the rejection of claim 40 should be reversed.

e. Claims 8, 9, 10, 11, 13, 14, 15, 16, and 17

Claims 8, 9, 10, 11, 13, 14, 15, 16, and 17 more specifically recite an apparatus for collecting a profile of a subroutine included in a program, or the features thereof.

Dependent claim 8 recites an apparatus as set forth in claim 12, wherein a plurality of storage units respectively corresponding to a plurality of executors of said subroutine are prepared; and said collecting section specifies said executor of said subroutine and stores said profile of said subroutine corresponding to said specified executor in said storage unit. Dependent claim 9 recites an apparatus as set forth in claim 8, wherein said collecting section individually stores profiles of a plurality of subroutines corresponding to a specified executor in said storage unit.

Dependent claim 10 recites an apparatus as set forth in claim 9, wherein said collecting section individually stores a first profile of a subroutine called by a main routine, and a second profile of the subroutine called by another subroutine, in said storage unit. Dependent claim 11 recites an apparatus as set forth in claim 10, wherein said collecting section stores said second profile and calling relationship information relating to said second profile, said calling relationship information indicating a relationship between said other subroutine and said called subroutine, in said storage unit.

Dependent claim 13 recites an apparatus as set forth in claim 12, wherein said collecting section stores times of calling of said subroutine corresponding to said branch destination

address as said profile in said storage unit. Dependent claim 14 recites an apparatus as set forth in claim 12, wherein said collecting section obtains an overhead of said subroutine as said profile and stores said overhead in said storage unit.

Dependent claim 15 recites an apparatus as set forth in claim 13, wherein said collecting section, when said identified result is said calling instruction, stores an identifier of an executor of said subroutine corresponding to said calling instruction and said branch destination address in said storage unit. Dependent claim 16 recites an apparatus as set forth in claim 12, wherein said collecting section, when said identified result is said calling instruction and said branch source address and said branch destination address are addresses of said subroutines, stores said branch source address and branch destination address as calling relationship information indicating a calling source subroutine and a calling destination subroutine in said storage unit, and stores at least one of the cumulative execution time and the times of calling in said calling destination subroutine in the call source subroutine, as said profile corresponding to said calling relationship information, in said storage unit.

Dependent claim 17 recites an apparatus as set forth in claim 12, further comprising a setting section setting an execution environment of a source of said interrupt so as to generate said interrupt when said branch instruction is executed during the execution of said program.

Even an *arguendo* combination of Alexander, Smolders, and Yeh does not explicitly disclose the claimed subject matter, and further, *prima facie* obviousness is not established for the reasons discussed herein in conjunction with claim 12. Therefore, it is submitted that dependent claims 8, 9, 10, 11, 13, 14, 15, 16, and 17 patentably distinguish over the prior art.

Thus, it is submitted that the rejection of claims 8, 9, 10, 11, 13, 14, 15, 16, and 17 is incorrect and the rejection of claims 8, 9, 10, 11, 13, 14, 15, 16, and 17 should be reversed.

f. Claims 19, 20, 21, 22, 24, 25, 26, 27, 28, and 41

Claims 19, 20, 21, 22, 24, 25, 26, 27, 28, and 41 are patentable over the cited art, for reasons similar to those discussed above respectively for claims 8, 9, 10, 11, 13, 14, 15, 16, 17, and 40, but specify a computer readable medium storing a program for a computer executing a process for collecting a profile of a subroutine included in a target program as set forth in claim 23.

As discussed above with reference to claim 12, claim 23, and claims 8, 9, 10, 11, 13, 14, 15, 16, 17, and 40, the rejection is incorrect since even an *arguendo* combination of Alexander, Smolders, and Yeh does not explicitly disclose the claimed subject matter, and further, *prima facie* obviousness is not established. Therefore, it is submitted that dependent claims 19, 20,

21, 22, 24, 25, 24, 25, 26, 27, 28, and 41 patentably distinguish over the prior art and the rejection is incorrect.

Thus, it is submitted that the rejection of claims 19, 20, 21, 22, 24, 25, 26, 27, 28, and 41 should be reversed.

g. Claims 30, 31, 32, 33, 35, 36, 37, 38, 39, and 42

Claims 30, 31, 32, 33, 35, 36, 37, 38, 39 and 42 are patentable over the cited art, for reasons similar to those discussed above respectively for claims 8, 9, 10, 11, 13, 14, 15, 16, 17, and 40, but specify a method for collecting a profile of a subroutine included in a program as set forth in claim 34.

Even an *arguendo* combination of Alexander, Smolders, and Yeh does not explicitly disclose the claimed subject matter, and further, *prima facie* obviousness is not established for the reasons discussed in conjunction with claim 12, claim 34, and claims 8, 9, 10, 11, 13, 14, 15, 16, 17, and 40. Therefore, it is submitted that dependent claims 30, 31, 32, 33, 35, 36, 37, 38, 39 and 42 patentably distinguish over the prior art and the rejection is incorrect

Thus, it is submitted that the rejections of claims 30, 31, 32, 33, 35, 36, 37, 38, 39 and 42 should be reversed.

Overall Summary

In view of the law and facts stated herein, the Appellant respectfully submits that the Examiner has failed to establish the obviousness rejection against the pending claims, and the Examiner's findings of unpatentability regarding claims 8-17, 19-28 and 30-42 should be reversed and the patentability over the presently cited references be affirmed.

The Commissioner is hereby authorized to charge any additional fees required in connection with the filing of this Appeal Brief to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: February 23, 2007

By _____
Paul W. Bobowiec
Registration No. 47,431

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

VIII: CLAIMS APPENDIX

1.-7. (CANCELED)

8. An apparatus according to claim 12, wherein a plurality of storage units respectively corresponding to a plurality of executors of said subroutine are prepared; and

said collecting section specifies said executor of said subroutine and stores said profile of said subroutine corresponding to said specified executor in said storage unit.

9. An apparatus according to claim 8, wherein said collecting section individually stores profiles of a plurality of subroutines corresponding to a specified executor in said storage unit.

10. An apparatus according to claim 9, wherein said collecting section individually stores a first profile of a subroutine called by a main routine, and a second profile of the subroutine called by another subroutine, in said storage unit.

11. An apparatus according to claim 10, wherein said collecting section stores said second profile and calling relationship information relating to said second profile, said calling relationship information indicating a relationship between said other subroutine and said called subroutine, in said storage unit.

12. An apparatus for collecting a profile of a subroutine included in a program, comprising:

a storage unit storing a profile;

an analyzing section, when an interrupt is generated by execution of a branch instruction during execution of said program:

obtaining a branch source address and a branch destination address from a

source of said interrupt, and

identifying a type of said branch instruction by obtaining an instruction code from said branch source address and decoding said instruction code; and

a collecting section:

obtaining said branch source address, said branch destination address, and an identified result from said analyzing section when the identified instruction is a calling instruction or a return instruction of said subroutine; and

when said identified result is said calling instruction, storing said branch destination address as a subroutine address corresponding to said calling instruction and a calling time of said subroutine corresponding to said calling instruction in said storage unit,

when said identified result is said return instruction,

obtaining a return time of said subroutine corresponding to said return instruction,

calculating a execution time of said subroutine based on said obtained return time and said calling time, and

storing a cumulative value of said execution time as said profile in correspondence with said branch destination address in said storage unit, and

when the identified branch instruction is neither a calling instruction nor a return instruction, said interrupt is terminated.

13. An apparatus according to claim 12, wherein said collecting section stores times of calling of said subroutine corresponding to said branch destination address as said profile in said storage unit.

14. An apparatus according to claim 12, wherein said collecting section obtains an overhead of said subroutine as said profile and stores said overhead in said storage unit.

15. An apparatus according to claim 13, wherein said collecting section, when said identified result is said calling instruction, stores an identifier of an executor of said subroutine corresponding to said calling instruction and said branch destination address in said storage unit.

16. An apparatus according to claim 12, wherein said collecting section, when said identified result is said calling instruction and said branch source address and said branch destination address are addresses of said subroutines, stores said branch source address and branch destination address as calling relationship information indicating a calling source subroutine and a calling destination subroutine in said storage unit, and stores at least one of the cumulative execution time and the times of calling in said calling destination subroutine in the call source subroutine, as said profile corresponding to said calling relationship information, in said storage unit.

17. An apparatus according to claim 12, further comprising a setting section setting an execution environment of a source of said interrupt so as to generate said interrupt when said branch instruction is executed during the execution of said program.

18. (CANCELED)

19. A computer readable medium according to claim 23, wherein said computer further executes the process by:

specifying an executor of said subroutine; and

storing said profile in a storage unit corresponding to said specified executor within a plurality of storage unit provided with each executor.

20. A computer readable medium according to claim 19, wherein said computer further executes the process by storing, when a specified executor executes a plurality of subroutines, profiles of said plurality of subroutines in said storage unit corresponding to said specified executor.

21. A computer readable medium according to claim 20, wherein said computer further executes the process of storing a first profile of said subroutine called by a main routine and a second profile of said subroutine called by another subroutine in said corresponding storage unit, regarding to each of said subroutines.

22. A computer readable medium according to claim 21, wherein said computer further executes the process by storing said second profile and calling relationship information relating to said second profile, said calling relationship information indicating a relationship between said other subroutine and said subroutine.

23. A computer readable medium a storing a program for a computer executing a process for collecting a profile of a subroutine included in a target program, by:

obtaining a branch source address and a branch destination address from a source of an interrupt when said interrupt is generated by execution of a branch instruction during execution of the target program;

identifying a type of said branch instruction by obtaining an instruction code from said branch source address and decoding said instruction code;

storing said branch destination address as a subroutine address corresponding to said calling instruction and a calling time of said subroutine corresponding to said calling instruction in a storage unit when said identified result is said calling instruction; and

when said identified result is said return instruction,

obtaining a return time of said subroutine corresponding to said return instruction,

calculating a execution time of said subroutine based on said obtained return time and said calling time, and

storing a cumulative value of said execution time as said profile in correspondence with said branch destination address in said storage unit, and

when the identified branch instruction is neither a calling instruction nor a return instruction, said interrupt is terminated.

24. A computer readable medium according to claim 23, wherein said computer further executes the process by storing times of calling of said subroutine corresponding to said branch destination address as said profile in said storage unit.

25. A computer readable medium according to claim 23, wherein said program further executes the process by storing an overhead of said subroutine as said profile in said storage unit.

26. A computer readable medium according to claim 23, wherein said computer further executes the process by storing, when said identified result is said calling instruction, an identifier of an executor of said subroutine corresponding to said calling instruction and said branch destination address in said storage unit.

27. A computer readable medium according to claim 23, wherein said computer further executes the process when said identified result is said calling instruction and said brunch source address and said branch destination address are addresses of said subroutines, storing said branch source address and branch destination address as calling relationship

information indicating a calling source subroutine, and a calling destination subroutine in said storage unit; and storing at least one of the cumulative execution time and the times of calling in said calling destination subroutine in the call source subroutine, as said profile corresponding to said calling relationship information, in said storage unit.

28. A computer readable medium according to claim 23, wherein said computer further executes the process by setting an execution environment of a source of said interrupt so as to generate said interrupt when said branch instruction is executed during the execution of said program.

29. (CANCELED)

30. A method according to claim 34, further comprising:

specifying an executor of the subroutine; and

storing the profile in a storage unit corresponding to the specified executor within a plurality of storage unit provided with each executor.

31. A method according to claim 30, further comprising storing, when a specified executor executes a plurality of subroutines, profiles of the plurality of subroutines in the storage unit corresponding to the specified executor.

32. A method according to claim 31, further comprising storing a first profile of the subroutine called out by a main routine and a second profile of the subroutine called out by another subroutine in the corresponding storage unit, regarding to each of the subroutines.

33. A method according to claim 32, further comprising storing the second profile and

calling relationship information relating to the second profile, the calling relationship information indicating a relationship between the other subroutine and the subroutine.

34. A method for collecting a profile of a subroutine included in a program, comprising:

obtaining a branch source address and a branch destination address from a source of an interrupt when the interrupt is generated by execution of a branch instruction during execution of said program;

identifying a type of the branch instruction by obtaining] an instruction code from the branch source address and decoding the instruction code;

storing the branch destination address as a subroutine address corresponding to the calling instruction and a calling time of the subroutine corresponding to the calling instruction in a storage unit when the identified result is the calling instruction; and

when the identified result is the return instruction,

obtaining a return time of the subroutine corresponding to the return instruction,

calculating a execution time of the subroutine based on the obtained return time and the calling time, and

storing a cumulative value of the execution time as the profile in correspondence with the branch destination address in the storage unit, and

when the identified branch instruction is neither a calling instruction nor a return instruction, said interrupt is terminated.

35. A method according to claim 34, further comprising storing times of calling of the subroutine corresponding to the branch destination address as the profile in said storage unit.

36. A method according to claim 34, further comprising storing an overhead of the

subroutine as the profile in the storage unit.

37. A method according to claim 34, further comprising storing, when the identified result is the calling instruction, an identifier of an executor of the subroutine corresponding to the calling instruction and the branch destination address in the storage unit.

38. A method according to claim 34, further comprising, when the identified result is the calling instruction and the branch source address and the branch destination address are addresses of the subroutines, storing the branch source address and the branch destination address as calling relationship information indicating a calling source subroutine, and a calling destination subroutine in said storage unit; and storing at least one of the cumulative execution time and the times of calling in the calling destination subroutine in the call source subroutine, as the profile corresponding to the calling relationship information, in the storage unit.

39. A method according to claim 34, further comprising setting an execution environment of a source of the interrupt so as to generate the interrupt when the branch instruction is executed during the execution of the program.

40. An apparatus according to claim 13, wherein the collecting section generates a control table corresponding to each executor of the subroutine on the storage unit,

wherein the control table includes an executor managing table, a subroutine managing table, and a calling managing table,

wherein the executor managing table stores an identifier of the executor and a pointer to assign the subroutine managing table,

wherein the subroutine managing table is generated for every subroutine executed by the executor, the subroutine managing table storing a subroutine address, times of calling of the

subroutine, a cumulative execution time of the subroutine, the last called time of the subroutine, and a pointer to assign the calling managing table, and

wherein the calling managing table is generated for every subroutine called by the subroutine, the calling managing table storing a branch source address as a calling subroutine address, a branch destination address as a called subroutine address, times of calling of the called subroutine, a cumulative execution time of the called subroutine, the last called time of the called subroutine, and a pointer to specify the subroutine managing table managing the calling subroutine.

41. A computer readable medium according to claim 23, wherein the computer further executes the process by generating a control table corresponding to each executor of the subroutine on the storage unit,

wherein the control table includes an executor managing table, a subroutine managing table, and a calling managing table,

wherein the executor managing tables stores an identifier of the executor and a pointer to assign the subroutine managing table,

wherein the subroutine managing table is generated for every subroutine executed by the executor, the subroutine managing table storing a subroutine address, times of calling of the subroutine, a cumulative execution time of the subroutine, the last called time of the subroutine, and a pointer to assign the calling managing table,

wherein the calling managing table is generated for every subroutine called by the subroutine, the calling managing table storing a branch source address as a calling subroutine address, a branch destination address as a called subroutine address, times of calling of the called subroutine, a cumulative execution time of the called subroutine, the last called time of the called subroutine, and a pointer to specify the subroutine managing table managing the calling subroutine.

42. A method according to claim 38, wherein the method further comprises generating a control table corresponding to each executor of the subroutine on the storage unit,

wherein the control table includes an executor managing table, a subroutine managing table, and a calling managing table,

wherein the executor managing table stores an identifier of the executor and a pointer to assign the subroutine managing table,

wherein the subroutine managing table is generated for every subroutine executed by the executor, the subroutine managing table storing a subroutine address, times of calling of the subroutine, a cumulative execution time of the subroutine, the last called time of the subroutine, and a pointer to assign the calling managing table, and

wherein the calling managing table is generated for every subroutine called by the subroutine, the calling managing table storing a branch source address as a calling subroutine address, a branch destination address as a called subroutine address, times of calling of the called subroutine, a cumulative execution time of the called subroutine, the last called time of the called subroutine, and a pointer to specify the subroutine managing table managing the calling subroutine.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.